

Using Binoculars for Fast Exploration and Map Construction in Chordal Graphs and Extensions

J  r  mie Chalopin, Emmanuel Godard and Antoine Naudin

LIF, Universit   Aix-Marseille and CNRS, FRANCE

Abstract. We investigate the exploration and mapping of anonymous graphs by a mobile agent. It is long known that, without global information about the graph, it is not possible to make the agent halt after the exploration except if the graph is a tree. We therefore endow the agent with *binoculars*, a sensing device that can show the local structure of the environment at a constant distance of the agent’s current location and investigate networks that can be efficiently explored in this setting.

In the case of trees, the exploration without binoculars is fast (i.e. using a DFS traversal of the graph, there is a number of moves linear in the number of nodes). We consider here the family of Weetman graphs that is a generalization of the standard family of chordal graphs and present a new deterministic algorithm that realizes Exploration of any Weetman graph, without knowledge of size or diameter and for any port numbering. The number of moves is linear in the number of nodes, despite the fact that Weetman graphs are not sparse, some having a number of edges that is quadratic in the number of nodes.

At the end of the Exploration, the agent has also computed a map of the anonymous graph.

Keywords: Mobile Agent, Graph Exploration, Map Construction, Anonymous Graphs, Linear Time, Chordal Graphs, Weetman graphs

Eligible for the Best Student Paper Award

1 Introduction

Mobile agents are computational units that can progress autonomously from place to place within an environment, interacting with the environment at each node that it is located on. These can be hardware robots moving in a physical world or software robots. Such software robots (sometimes called bots, or agents) are already prevalent in the Internet, and are used for performing a variety of tasks such as collecting information or negotiating a business deal. More generally, when the data is physically dispersed, it can be sometimes beneficial to move the computation to the data, instead of moving all the data to the entity performing the computation. The paradigm of mobile agent computing / distributed robotics is based on this idea. As underlined in [Das13], the use of mobile agents has been advocated for numerous reasons such as robustness against network disruptions, improving the latency and reducing network load,

providing more autonomy and reducing the design complexity, and so on (see e.g. [LO99]).

For many distributed problems with mobile agents, exploring, that is visiting every location of the whole environment, is an important prerequisite. In its thorough exposition about Exploration by mobile agents [Das13], S. Das presents numerous variations of the problem. In particular, it can be noted that, given some global information about the environment (like its size or a bound on the diameter), it is always possible to explore, even in environments where there is no local information that enables to know, arriving on a node, whether it has already been visited (e.g. anonymous networks). If no global information is given to the agent, then the only way to perform a network traversal is to use an *unlimited* traversal (e.g. with a classical BFS or Universal Exploration Sequences [AKL⁺79,Kou02,Rei08] with increasing parameters). This infinite process is sometimes called *Perpetual Exploration* when the agent visits infinitely many times every node. Perpetual Exploration has application mainly to security and safety when the mobile agents are a way to regularly check that the environment is safe. But it is important to note that in the case where no global information is available, it is impossible to always detect when the Exploration has been completed. This is problematic when one would like to use the Exploration algorithm composed with another distributed algorithm. In this note, we focus on fast Exploration with termination. It is known that in general anonymous networks, the only topology that enables to stop after the exploration is the tree-topology. From standard covering and lifting techniques, it is possible to see that exploring with termination a (small) cycle would lead to halt before a complete exploration in huge cycles. Moreover, using a simple DFS traversal, Exploration on trees has cover time that is linear in the number of nodes.

We have shown in [CGN15] that it is possible to explore, with full stop, non-tree topologies without global information using some local information. The information that is provided can be informally described as giving *binoculars* to the agent. This constant range sensor enables the agent to “see” the graph (with port numbers) that is induced by the adjacent nodes of its current location. See Section 3 for a formal definition.

Using binoculars is a quite natural enhancement for mobile robots. In some sense, we are trading some a priori global information (that might be difficult to maintain efficiently) for some local information that the agent can *autonomously* and *dynamically* acquire.

In [CGN15], a complete characterization of which networks can be explored with binoculars is given and the exploration time is proven to be not practicable for the whole family of networks that can be explored with binoculars. Here we focus on families that can be explored in a fast way, typically in a time linear in the number of nodes.

Chordal graphs are tree-like graphs where “leaves” are so-called simplicial vertices, *i.e.* vertices whose neighbourhood is a clique. Using binoculars, it is possible to locally detect such simplicial vertices. But how to leverage such detection to

get an exploration algorithm in anonymous graphs is not straightforward since it is not possible to mark nodes.

Our Results. We present an algorithm that efficiently explores all chordal graphs in a linear number of moves by a mobile agent using binoculars. The main contribution is that the exploration is fast even if the agent does not know the size or the diameter (or bounds). The algorithm actually leverages properties of chordal graphs that are verified in a larger class of graphs: the Weetman graphs. This family has been introduced by Weetman [Wee94] and can be defined with metric local conditions (see later). It contains the family of Johnson graphs.

Using binoculars, we therefore show it is possible to explore and map with halt dense graphs (having a number of edges quadratic in the number of nodes) in $O(n)$ moves, for any port numbering.

Related works. To the best of our knowledge, efficient Exploration using binoculars has never been considered for mobile agent on graphs. When the agent can only see the label and the degree of its current location, it is well-known that any Exploration algorithm can only halt on trees and a standard DFS algorithm enables to explore any tree in $O(n)$ moves. Gasieniec et al. [AGP⁺11] presented an algorithm that can explore any tree with a memory of size $O(\log n)$. For general anonymous graphs, Exploration with halt has mostly been investigated assuming at least some global bounds, in the goal of optimizing the move complexity. It can be done in $O(\Delta^n)$ moves using a DFS traversal while knowing the size n when the maximum degree is Δ . This can be reduced to $O(n^3 \Delta^2 \log n)$ using Universal Exploration Sequences [AKL⁺79, Kou02] that are sequences of port numbers that an agent can sequentially follow and be assured to visit any vertex of any graph of size at most n and maximum degree at most Δ . Reinhold [Rei08] showed that universal exploration sequences can be constructed in logarithmic space.

Trade-offs between time and memory for exploration of anonymous tree networks has been presented in [AGP⁺11]. Note that in this case, the knowledge of the size is required to halt the exploration. For example, there is a very simple exploration algorithm for cycles (“go through the port you are not coming from”) that needs the knowledge of the size to be able to halt. Here we are looking for algorithm that does not use (explicitly or implicitly) such knowledge.

Trading global knowledge for structural local information by designing specific port numbering, or specific node labels that enable easy or fast exploration of anonymous graphs have been proposed in [CFI⁺05, GR08, Ilc08]. Note that using binoculars is a local information that can be locally maintained contrary to the schemes proposed by these papers where the local labels are dependent of the full graph structure.

See also [Das13] for a detailed discussion about Exploration using other mobile agent models (with pebbles for examples).

2 Exploration with Binoculars

2.1 The Model

Mobile Agents. We use a standard model of mobile agents, that we now formally describe. A mobile agent is a computational unit evolving in an undirected simple graph $G = (V, E)$ from vertex to vertex along the edges. A vertex can have some label attached to it. There is no global guarantee on the labels, in particular vertices have no identity (anonymous/homonymous setting), i.e., local labels are not guaranteed to be unique. The vertices are endowed with a port numbering function available to the agent in order to let it navigate within the graph. Let v be a vertex, we denote by $\delta_v : V \rightarrow \mathbb{N}$, the injective port numbering function giving a locally unique identifier to the different adjacent nodes of v . We denote by $\delta_v(w)$ the port number of v leading to the vertex w , i.e., corresponding to the edge $vw \in E(G)$. We denote by (G, δ) the graph G endowed with a port numbering $\delta = \{\delta_v\}_{v \in V(G)}$.

When exploring a network, we would like to achieve it for any port numbering. So we consider the set of every graph endowed with a valid port numbering function, called \mathcal{G}^δ . By abuse of notation, since the port numbering is usually fixed, we denote by G a graph $(G, \delta) \in \mathcal{G}^\delta$.

The behaviour of an agent is cyclic: it obtains local information (local label and port numbers), computes some values, and moves to its next location according to its previous computation. We also assume that the agent can back-track, that is the agent knows via which port number it accessed its current location. We do not assume that the starting point of the agent (that is called the *homebase*) is marked. All nodes are a priori indistinguishable except from the degree and the label. We assume that the mobile agent is a Turing machine (with unbounded local memory). Moreover we assume that an agent accesses its memory and computes instructions instantaneously. An execution ρ of an algorithm \mathcal{A} for a mobile agent is composed by a (possibly infinite) sequence of moves by the agent. The length $|\rho|$ of an execution ρ is the total number of moves.

2.2 The Exploration Problem

We consider the classical exploration Problem for a mobile agent. An algorithm \mathcal{A} is an exploration algorithm if for any graph $G = (V, E)$ with binocular labelling, for any port numbering δ_G , starting from any arbitrary vertex $v_0 \in V$, the agent visits every vertex at least once and terminates.

We say that a graph G is *explorable* if there exists an Exploration algorithm that halts on G starting from any point. An algorithm \mathcal{A} explores a family of graphs \mathcal{F} if it is an Exploration algorithm such that for all $G \in \mathcal{F}$, \mathcal{A} halts and for all $G \notin \mathcal{F}$, either \mathcal{A} halts and explores G , either \mathcal{A} never halts; we say that it is a universal exploration algorithm for \mathcal{F} . We require the Exploration algorithm to not use any metric information about the graph (like the size).

3 Definitions and Notations

3.1 Graphs

We always assume simple and connected graphs. The following definitions are standard [Ros00]. Let G be a graph, we denote $V(G)$ (resp. $E(G)$) the set of vertices (resp. edges). If two vertices $u, v \in V(G)$ are adjacent in G , the edge between u and v is denoted by uv .

Loops, Paths and Cycles. A loop in a graph G is a sequence of vertices $(v_0, \dots, v_k) \subseteq V(G)$ such that either $v_i v_{i+1} \in E(G)$, either $v_i = v_{i+1}$, for every $0 \leq i < k$. The length of a loop is equal to the number of vertices composing it. A path p in a graph G is a loop (v_0, \dots, v_k) such that $v_i v_{i+1} \in E(G)$ for every $0 \leq i < k$. We say that the length of a path p , denoted by $|p|$, is the number of edges composing it. We denote by p^{-1} the inverted sequence of p . A path is simple if for any $i \neq j$, $v_i \neq v_j$. A cycle is a path such that $v_0 = v_k$, $k \in \mathbb{N}$. A cycle is simple if the path (v_0, \dots, v_{k-1}) is simple or it is the empty path. On a graph endowed with a port numbering, a path $p = (v_0, \dots, v_k)$ is labelled by $\lambda(p) = (\delta_{v_0}(v_1), \delta_{v_1}(v_2), \dots, \delta_{v_{k-1}}(v_k))$.

The distance between two vertices v and v' in a graph G is denoted by $d_G(v, v')$. It is the length of the shortest path between v and v' in G . The set $\text{pred}_{v_0}(v) = \{u \mid uv \in E(G) \wedge d(v_0, u) = d(v_0, v) - 1\}$ is called the set of predecessors of v . We denote it by $\text{pred}(v)$ if the context permits it.

We define $N_G(v, k)$ to be the subset of vertices of G at distance at most k from the vertex v in G . We define $B_G(v, k)$ to be the subgraph of G induced by $N_G(v, k)$.

Let p be a path in a graph G leading from a vertex v to w . We define $\text{DEST}_G : V(G) \times \mathbb{N}^{\mathbb{N}}$ such that $\text{DEST}_G(v, \lambda(p)) = w$, that is, $\text{DEST}_G(v, \lambda(p))$ is the vertex in G reached by the path labelled by $\lambda(p)$ starting from v_0 .

Layering partition and Clusters. A *layering* of a graph $G = (V, E)$ having a distinguished vertex v_0 is a partition of V into sphere $S^i = \{v \mid d(v_0, v) = i\}$, $\forall i = 1, 2, \dots$. A *layering partition* of G is a partition of each S^i into *clusters* C_1^i, \dots, C_p^i such that for every two vertices $u, v \in S^i$, u and v belong to C_j^i if and only if there is a path p from u to v passing inside S^i . We denote by $d(v_0, C)$ the distance h between vertices in C and v_0 .

We define below $\text{Cluster}(G)$, the graph of clusters of a graph G .

Definition 3.1. $\text{Cluster}(G) = (V, E)$ such that

- $V = \{\text{cluster } C \text{ of } G\}$
- $E = \{CC' \mid \exists v \in V(C), \exists v' \in V(C') \text{ and } vv' \in E(G)\}$

The set of *predecessors* of a cluster C in G , denoted by $\text{pred}(C)$, is composed by every cluster C' in G such that $C'C \in E(\text{Cluster}(G))$ and $d(v_0, C) = d(v_0, C') + 1$. Respectively, the set of *successors* of C , denoted by $\text{succ}(C)$ is composed by every component C' such that $CC' \in E(\text{Cluster}(G))$ and $d(v_0, C') = d(v_0, C) + 1$.

Binoculars. Our agent can use “binoculars” of range 1, that is, located on a vertex u , it can “see” the induced ball $B(u, 1)$ (with port numbers) of radius 1 centered on u . To formalize what the agent sees with its binoculars, we always assume that every graph G is endowed with an additional vertex labelling ν , called *binoculars labelling* such that for any vertex $v \in V(G)$, $\nu(v)$ is a graph isomorphic to $B(v, 1)$ endowed with a port numbering τ induced from G . Moreover, the agent is endowed with a primitive called `getBino()` permitting it to access to the binoculars labelling $\nu(u)$ of the vertex u currently visited, that is, located on u , `getBino()` returns $B(u, 1)$. Note that the agent knows which is the explored vertex u in $B(u, 1)$.

Coverings. We now present the formal definition of graph homomorphisms that capture the relation between graphs that locally look the same in our model. A map $\varphi : V(G) \rightarrow V(H)$ from a graph G to a graph H is a *homomorphism* from G to H if for every edge $uv \in E(G)$, $\varphi(u)\varphi(v) \in E(H)$. A homomorphism φ from G to H is a *covering* if for every $v \in V(G)$, $\varphi|_{N_G(v)}$ is a bijection between $N_G(v)$ and $N_H(\varphi(v))$.

This standard definition is extended to labelled graphs $(G, \delta, \text{label})$ and $(G', \delta', \text{label}')$ by adding the conditions that $\text{label}'(\varphi(u)) = \text{label}(u)$ for every $u \in V(G)$ and that $\delta_u(v) = \delta'_{\varphi(u)}(\varphi(v))$ for every edge $uv \in E(G)$. We have the following equivalent definition when G and G' are endowed with a port numbering.

Proposition 3.2. *Let $(G, \delta, \text{label})$ and $(G', \delta', \text{label}')$ be two labelled graphs, an homomorphism $\varphi : G \rightarrow G'$ is a covering if and only if*

- for all $u \in V(G)$, $\text{label}(u) = \text{label}'(\varphi(u))$,
- for all $u \in V(G)$, u and $\varphi(u)$ have same degree.
- for any $u \in V(G)$, for any $v \in N_G(u)$, $\delta_u(v) = \delta'_{\varphi(u)}(\varphi(v))$.

Proposition 3.3 (Universal Cover). *For any graph G , there exists a possibly infinite graph (unique up to isomorphism) denoted \hat{G} and a covering $\mu : \hat{G} \rightarrow G$ such that, for any graph G' , for any covering $\varphi : G' \rightarrow G$, there exists a covering $\gamma : \hat{G} \rightarrow G'$ and $\varphi \circ \gamma = \mu$.*

It is also possible to have a notion of simplicial covering. A *graph covering* $\varphi : G \rightarrow G'$ is a simplicial covering $\varphi : V(G) \rightarrow V(G')$ such that for any vertex $v \in V(G)$, $\nu(v) = \nu(\varphi(v))$. This notion captures the indistinguishable graphs for an agent endowed with Binoculars. We get the following definition for the simplicial universal cover,

Proposition 3.4 (Simplicial Universal Cover). *For any graph G , there exists a possibly infinite graph (unique up to isomorphism) denoted \hat{G} and a simplicial covering $\mu : \hat{G} \rightarrow G$ such that, for any graph G' , for any simplicial covering $\varphi : G' \rightarrow G$, there exists a simplicial covering $\gamma : \hat{G} \rightarrow G'$ and $\varphi \circ \gamma = \mu$.*

From standard distributed computability results [YK96, BV01, CGM12], it is known that the structure of the covering maps explains what can be computed or not. So in order to investigate the structure induced by coverings of graphs with

binoculars labelling, we will investigate the structure of simplicial coverings. We call simply “coverings” the simplicial coverings.

Note that the simplicial universal cover (as a graph with binoculars labelling) can differ from its universal cover (as a graph without labels). Consider for example, the triangle network.

Homotopy. We say that two loops $c = (v_0, v_1, \dots, v_{i-1}, v_i, v_{i+1}, \dots, v_k)$ and $c' = (v_0, v_1, \dots, v_{i-1}, v_{i+1}, \dots, v_k)$ in a graph G are related by an *elementary homotopy* if one of the following conditions holds (definitions from [BH99]):

- (Contracting) $v_i = v_{i+1}$,
- (Backtracking) $v_{i-1} = v_{i+1}$,
- (Pushing across a 2-cell) $v_{i-1}v_{i+1}$ is an edge of G .

Note that being related by an elementary homotopy is a reflexive relation (we can either increase or decrease the length of the loop). We say that two loops c and c' are homotopic equivalent if there is a sequence of loops c_1, \dots, c_k such that $c_1 = c$, $c_k = c'$, and for every $1 \leq i < k$, c_i is related to c_{i+1} by an elementary homotopy. A loop is k -contractible (for $k \in \mathbb{N}$) if it can be reduced to a vertex by a sequence of k elementary homotopies. A loop is contractible if there exists $k \in \mathbb{N}$ such that it is k -contractible.

Simple Connectivity. A *simply connected* graph is a graph where every loop can be reduced to a vertex by a finite sequence of elementary homotopies.

This definition is the graph version of the simplicial covering defined in [CGN15] for simplicial complexes. Simply connectivity have a lot of interesting combinatorial and topological properties. In our proofs, we rely on the following fundamental result below. Even if this results applied for simplicial complexes, we can prove that it holds for graphs and simplicial graph covering as defined above.

Proposition 3.5 ([LS77]). *Let (G, ν) be a connected graph with binoculars labelling, then G is isomorphic to \hat{G} , the universal simplicial covering of G if and only if G is simply connected.*

In fact, in order to check the simple connectivity of a graph G , it is enough to check that all its simple cycles are contractible. The proof is straightforward and presented in Appendix. We get the following Proposition,

Proposition 3.6. *A graph G is simply connected if and only if every simple cycle is contractible.*

In Figure 1, we present two examples of simplicial covering maps, φ is from the universal cover, and φ' shows the general property of coverings that is that the number of vertices of the bigger graph is a multiple (here the double) of the number of vertices of the smaller graph.

Weetman Graphs. We present now the family of graphs investigated in this paper.

Definition 3.7 (Weetman Graphs [Wee94]). *A graph G is Weetman if it satisfies for every vertex v_0 the following properties:*

Fig. 1: Simplicial Covering

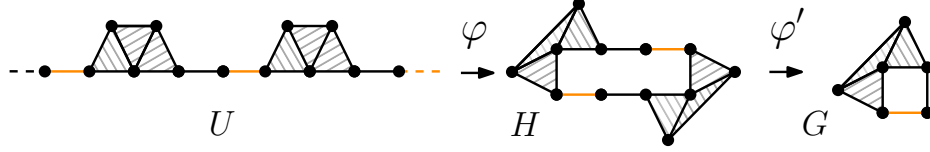
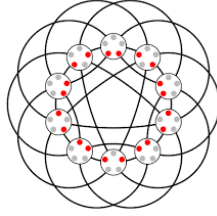


Fig. 2: Johnson Graph $J(5, 2)$



(TRI) **Triangle Condition.** for every adjacent nodes $v, v' \in V(G)$ at distance k from v_0 , there is a vertex u at distance $k - 1$ from v_0 such that $uv, uv' \in E(G)$.

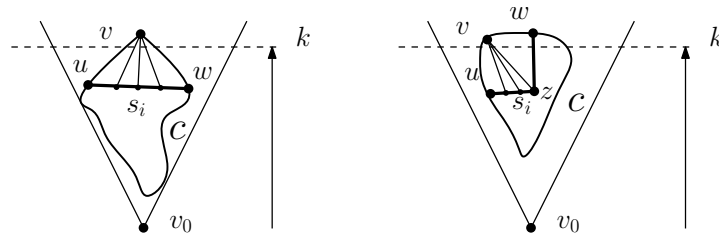
(INT) **Interval Condition.** For every $v \in V(G)$, the subgraph induced by $\text{pred}_{v_0}(v)$, the predecessors of v , is connected.

The family of Weetman graphs contains chordal graphs but also non-chordal graphs like Johnson graphs [RT87]. Johnson graphs are graphs whose vertices are subset of k elements of a set with n elements, and whose edges link subsets that can be obtained from one another by removing and adding one element (see Fig. 2 for instance).

They belong to the family of Simply connected graphs.

4 Properties of Weetman Graphs

Fig. 3: Illustration of different cases of the proof of Lemma 4.1



Proposition 4.1. *Weetman graphs are simply connected*

Proof. Suppose by contradiction that there is one not contractible cycle c in a Weetman graph G . Among all not contractible cycle c' in G , choose the cycle c minimising $k = d(v_0, c) = \max\{d(v_0, v') \mid v' \in c\}$ and minimising $|c \cap S^k|$.

We prove that there is another cycles minimising either $d(v_0, c)$, either $|c \cap S^k|$ which is not contractible in G , contradicting our hypothesis.

There are two cases, if $|c \cap S^k| = 1$, then there are $u, v, w \in V(c)$ such that $u, w \in S^{k-1}$, $v \in S^k$, $uv, vw \in E(G)$. Moreover, Since c minimises $d(v_0, c)$, there is no edge $uw \in E(G)$.

Since G is Weetman, from Condition INT applied on the triple of vertices u, v, w , there is a path $p = (s_1 \dots s_n) \subset S^{k-1}$ such that $s_1 = u, s_n = w$ and for every $1 \leq i \leq n$, $s_i v \in E(G)$.

Since the cycle $c' = c \setminus \{uvw\} \cup \{p\}$ is related to c by a sequence of elementary homotopies, c' is also not contractible and since $d(v_0, c') = k - 1$. We get a contradiction on the choice of c .

If $|c \cap S^k| > 1$, let $u \in S^{k-1}$, $v, w \in S^k$ such that u and w are the previous and consecutive vertex of v in c , i.e. $uv, vw \in E(c)$.

From Condition TRI (on the edge vw), there is a vertex $z \in S^{k-1}$ such that $zvw \in E(G)$ is a triangle.

From Condition INT (on the triple of vertices uvw), there is a path $q = \{s_0 \dots s_\ell\} \subset S^{k-1}$ such that $s_0 = u, s_\ell = z$ and $s_i v \in E(G), \forall 0 \leq i \leq \ell$.

Consequently, there is a cycle $c' = c \setminus \{uvw\} \cup \{qw\}$ passing through vertices uqw instead of uvw . Moreover, since c' is related to c by a sequence of elementary homotopies, c' is also not contractible.

Consequently, since $|c' \cap S^k| < |c \cap S^k|$ and c' is not contractible, we get a contradiction on the choice of c . □

The following Lemma prove that clusters in a simply connected graph have a particular topology that permit us later in this document to explore Weetman Graphs in a quasi optimal way. Since the proof uses combinatorial tools which are not needed in the remaining part, the proof is left to the appendix.

Theorem 1. *Clusters of a simply connected graph form a tree*

Consequently, from Theorem 6, we get trivially the next corollary.

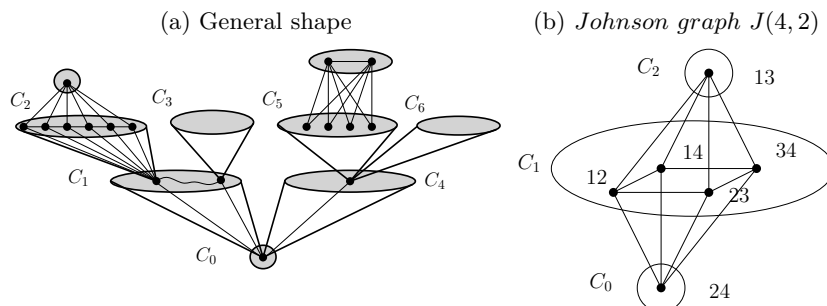
Corollary 4.2. *The clusters of a Weetman graph form a tree.*

Note that this tree can be reduced to a path in some cases, like Johnson graphs or triangulated sphere. See Fig. 4.

From Corollary 4.2, every cluster $C \subset S^k$ of a Weetman graph G admits a unique ancestor cluster, called $a(C)$, in the graph. Note that $a(C) \subset S^{k-1}$ by definition and for the cluster C_0 containing v_0 , $\text{pred}(C_0) = C_0$.

This particularity of Simply connected graph are the basis of the following exploration algorithm. As illustrated in the next corollary, such a property eases the exploration and enables a "quasi" optimal exploration algorithm.

Fig. 4: Clusters in a Weetman graph



Corollary 4.3. *Every path leading from the homebase $v_0 \in V(G)$ of the agent to a connected cluster C goes through $a(C)$.*

5 Weetman-universal Exploration Algorithm

Exploring in anonymous graph is difficult. Because of the lack of ids attached to a node, it might not be possible to know whether a node where the agent is located is actually new. We introduce the following terminology. A node is *explored* if the agent has already been to this node. A node is *discovered* if it has been seen from another node (that is, if it is adjacent to an explored node). In the following, we give a description of Algorithm 1. In a nutshell, by a DFS traversal of the tree of clusters, for every cluster C , the agent will explore all nodes of C , discovering in this way all child clusters of C . In this way, the agent is able to explore clusters in a DFS fashion.

Algorithm 1 is divided into phases. Between phases, the agent navigates the tree of clusters in a DFS fashion using a stack. In each phase, the agent explores a cluster and updates local structures (more details below). At the end of the phase, the agent extends its map with the new identified vertices and corresponding edges. From the updated map, the agent computes the new clusters that appeared in MAP. We now give more details on the computations.

The map Map. The map computed by the agent is denoted by MAP. A vertex $v \in V(G)$ is represented in $V(\text{MAP})$ by a unique integer n when the agent identifies the vertex v . The port numbering of map MAP is induced by the port numbering δ of the network G .

Local structures for the identification. Let p be the path followed by the agent from the beginning of the exploration starting at v_0 . Let u be the current location vertex of the agent in G corresponding to n in its map MAP and let n_0 be the vertex corresponding to the homebase of the agent in MAP. That is, $\text{DEST}_G(v_0, \lambda(p)) = u$ and $\text{DEST}_{\text{MAP}}(n_0, \lambda(p)) = n$.

First, remark that the agent exploring the graph knows in its map on which vertex $n = \text{DEST}_{\text{MAP}}(n_0, \lambda(p))$ it is located.

During the exploration of a cluster C , for every explored vertex $u \in V(G)$ identified by $n \in V(\text{MAP})$, the agent looks through its binoculars, that is, it calls `getBino()` and access to $B_G(u, 1)$, the ball of radius 1 around u (Line 13).

The ball $B_G(u, 1)$ obtained is stored into the set \mathcal{B} in order to, at the end of the phase, update and verify the "local" correctness of the map MAP . We denote by $\mathcal{B}[n]$ the ball obtained after calling `getBino()` on a vertex $n \in V(\text{MAP})$. Since the agent have to know which vertex in $\mathcal{B}[n]$ corresponds to its current location, we introduce $\psi(n) \in \mathcal{B}[n]$ to denote the vertex corresponding to n in $\mathcal{B}[n]$.

At the end of the phase, i.e., the end of the exploration of the cluster, the agent, using \mathcal{B} , updates two data structures that are used to identify the nodes and update MAP .

The first structure **PRE-VERT** encodes the existence of new vertices that are not present in MAP as seen by the agent from an explored node n at the phase i .

Since such new vertices are linked to a vertex explored phase i , we encode a vertical edge by a tuple (n, p, q) where n is the id of the explored vertex and (p, q) is the labelling of the vertical edge. We call *pre-vertex* a pair (n, p) . Pre-vertices give us all newly discovered vertices. Note that q is stored along the pre-vertex (n, p) only to simplify the computation of MAP at the end of the phase.

The main idea to correctly map newly discovered vertices is, at the end of a phase, to find all vertical edges pointing to this node. So, we add a second structure, denoted by \mathcal{R}_{\equiv} , encoding the elementary relation between two pre vertices corresponding to a same vertex in G . So if there is $((n, p), (m, q)) \in \mathcal{R}_{\equiv}$, then there is a couple of pre-vertices $(n, p), (m, q) \in \text{PRE-VERT}$ such that n and m are the ids of vertices explored during the phase, nm is an edge of $E(\text{MAP})$ and there is $uvw \subseteq \mathcal{B}[n]$ such that $u = \psi(n)$, $\lambda(nm) = \lambda(uv)$ and $\text{DEST}_{\text{MAP}}(n, \delta_u(w))$ is not defined in MAP .

In order to update MAP in such a way that all edges between discovered nodes are correctly mapped, we have to distinguish two kind of edges. There are edges between an explored node and a newly discovered node. They are called "vertical" edges. But edges between two discovered nodes are also to be correctly mapped. These edges are called "horizontal" edges.

To gather "vertical" edges, \equiv is sufficient as explained below. To gather "horizontal" edges, the set **Hor** is introduced. Since it is not possible to identify "horizontal" edges before the end of the exploration of the cluster, we store in **Hor**, together with the port numbers associated to the horizontal edge, only the pre-vertices. Namely, elements of **Hor** are tuples $(n, p_1, p_2, (r, s))$ such that there is a triangle uvw in $\mathcal{B}[n]$ where $u = \psi(n)$ and there is two pre vertices $(n, p_1), (n, p_2) \in \text{PRE-VERT}$ such that $\delta_u(v) = p_1$, $\delta_u(w) = p_2$ and $\lambda(vw) = (r, s)$. Note that by definition, $\text{DEST}_{\text{MAP}}(n, p_1)$ and $\text{DEST}_{\text{MAP}}(n, p_2)$ are not defined in MAP during the exploration of the cluster.

Updating the map. First remark that the transitive and reflexive closure of \equiv , denoted by \equiv^* , is an equivalence relation between pre vertices (it is straight-

forward that \equiv is reflexive). We denote by $[n, p]$, the representative (or the equivalence class) of a class of pre-vertices including (n, p) via \equiv^* .

To identify new nodes (Line 26), we compute the quotient of the relation \equiv^* over pre-vertices stored in **PRE-VERT**. Then, for every equivalence classes $[n, p]$ of pre-vertices that arises, we add a new vertex $\overline{[(n, p)]}$ in **MAP**. Moreover, every node $n \in V(\text{MAP})$ is endowed with an additional label **Cluster**(n) $\in \mathbb{N}$ to store the identity of the cluster including n .

Then, for every pre vertex $(n, p, q) \in \text{PRE-VERT}$, we add a "vertical" edge linking n to $\overline{[(n, p)]}$ labelled by (p, q) if the edge is not already present in **MAP**.

To update the "horizontal" edges of **MAP**, for every $(n, p_1, p_2, (r, s)) \in \text{Hor}$, we add an edge between $\overline{[(n, p_1)]}$ and $\overline{[(n, p_2)]}$ labelled by (r, s) if the edge is not already present in **MAP**.

Additionally, if the agent ends phase i , for every vertex $n \in V(\text{MAP})$ explored during this phase, **Vis**(n) is set to i .

Once **MAP** has been updated, we compare the map obtained and what we saw during the exploration of the cluster. That is, for every vertex u corresponding to n explored phase i , we compare $B_{\text{MAP}}(n, 1)$ and $\mathcal{B}[n]$, the binoculars labelling obtained from u . If we detect an error in the map, we decide to continue the exploration forever in order to respect the exploration specification. This case will be more discussed later in this document. If no error is detected, the new clusters that appear in **MAP** are computed, numbered, and push to the stack **STACK** at Line 33.

The agent stops its exploration when it remains no cluster to explore, that is, when **STACK** is empty.

Remark 5.1. *The only cluster at level 0 is composed of the homebase of the agent v_0 .*

Let $\partial \text{MAP} = \{n \in V(\text{MAP}) \mid \text{Vis}(n) = \perp\}$ be the subgraph induced from G composed by the set of vertices in G discovered and not yet explored by the agent.

Note that at MAP^i corresponds to the map computed at the end of the phase i .

First we prove the correctness properties, that is, if no error is detected and if the algorithm halts then the graph is explored. Then, the proof of the termination of Algorithm 1 on Weetman graphs will be straightforward.

The core of correctness proof is the following theorem

Theorem 2. *For every phase i of the algorithm, there is an homomorphism $\varphi : M^i \rightarrow G$ such that*

- for every $n \in V(\text{MAP}^i)$, $\varphi|_{N_{M^i}(n)}$ is injective
- for every $n \in V(\text{MAP}^i \setminus \partial \text{MAP}^i)$, $\varphi|_{N_{M^i}(n)}$ is surjective

Theorem 2 is proved by an induction on the phases perform by the agent during the execution.

The homomorphism φ is based on the following corollary of Theorem 2,

Algorithm 1: Weetman Graphs Exploration

```

1 Main Procedure
2   Add  $\mathbf{n}_0$  to  $V(\text{MAP})$ 
3    $\text{STACK} \leftarrow \text{Cluster}(\mathbf{n}_0) \leftarrow \text{Vis}(\mathbf{n}_0) \leftarrow 0$  /* homebase cluster */
4
5   while  $\text{STACK} \neq \emptyset$  do
6     /* Beginning of the phase  $i$  */
7      $\mathbf{n}_c \leftarrow \text{top element of STACK}$ 
8      $\mathcal{C} \leftarrow \{\mathbf{n} \in V(\text{MAP}) \mid \text{Cluster}(\mathbf{n}) = \mathbf{n}_c\}$ 
9     Compute a shortest path  $\text{PATH}$  which from the current location  $\mathbf{m}$  of the
10    agent, explores the cluster  $\mathcal{C}$ 
11
12    forall the  $\mathbf{n} \in \mathcal{C}$  /* Cluster exploration */
13    do
14      Go to the vertex  $\mathbf{n}$  following  $\text{PATH}$ 
15       $\text{Vis}(\mathbf{n}) = i$ 
16       $\mathcal{B}[\mathbf{n}] \stackrel{\cup}{\leftarrow} \text{getBino}(\mathbf{n})$  /* Looking thought Binoculars */
17
18      forall the  $\mathbf{n} \in \mathcal{C}$  do
19        Get  $\mathcal{B}[\mathbf{n}]$  from  $\mathcal{B}$  and let  $u \in V(\mathcal{B}[\mathbf{n}])$  corresponding to  $\mathbf{n}$  /* new pre vertices */
20
21        forall the  $uw \in E(\mathcal{B}[\mathbf{n}])$  s.t. there is no adjacent edge to  $\mathbf{n}$  in  $\text{MAP}$ 
22        labelled by  $\lambda(uw) = (\delta_u(w), \delta_w(u))$  do
23           $\text{PRE-VERT} \stackrel{\cup}{\leftarrow} (\mathbf{n}, \delta_u(w), \delta_w(u))$ 
24
25          forall the triangle  $uvw \subseteq \mathcal{B}[\mathbf{n}]$  do
26            /* new pre vertices/ vertical relation */
27            if there is an edge labelled  $\lambda(uv)$  and there is no edge labelled
28             $\lambda(uw)$  and  $\lambda(vw)$  adjacent to  $\mathbf{n}$  in  $\text{MAP}$  then
29              Let  $m$  be the vertex in  $\text{MAP}^i$  such that  $nm \in E(\text{MAP})$  and
30               $\lambda(nm) = \lambda(uv)$ 
31               $\equiv \stackrel{\cup}{\leftarrow} ((\mathbf{n}, \delta_u(w), \delta_w(u)), (\mathbf{m}, \delta_v(w), \delta_w(v)))$ 
32
33              /* new horizontal edge relation */
34              if there is no edge labelled by  $\lambda(uv), \lambda(uw)$  adjacent to  $\mathbf{n}$  in  $\text{MAP}$ 
35              then
36                 $\text{Hor} \stackrel{\cup}{\leftarrow} (\mathbf{n}, \delta_u(v), \delta_u(w), (\delta_v(w), \delta_w(v)))$ 
37
38          /* Updating MAP */
39
40        forall the  $[\mathbf{n}, p, q] \in \text{PRE-VERT} / \equiv$  do
41          Add a new vertex  $\overline{[(\mathbf{n}, p)]}$  to  $\text{MAP}$ 
42           $\text{Vis}(\overline{[(\mathbf{n}, p)]}) = \perp$ 
43
44        forall the  $(\mathbf{n}, p, q) \in \text{PRE-VERT}$  do
45          Add a new edge  $\mathbf{n}[\overline{[(\mathbf{n}, p)]}]$  labelled  $(p, q)$  to  $\text{MAP}$ 
46
47        forall the  $(\mathbf{n}, p, q, (p', q')) \in \text{Hor}$  do
48          Add a new edge  $\overline{[(\mathbf{n}, p)]}[\overline{[(\mathbf{n}, q)]}]$  labelled  $(p', q')$  in  $\text{MAP}$ 
49
50        if  $\forall \mathbf{n} \in \mathcal{C}, \mathcal{B}[\mathbf{n}]$  is isomorphic to  $B_{\text{MAP}}(\mathbf{n}, 1)$  then
51          Push in  $\text{STACK}$  the new clusters in  $\text{MAP}$ 
52          For every new vertex  $\overline{[(\mathbf{n}, p)]}$ , update  $\text{Cluster}(\overline{[(\mathbf{n}, p)]})$ 
53        else
54          Continue forever the execution along an edge
55
56      /* End of the phase  $i$  */

```

Corollary 5.2. *For every phase i , For every vertex $n \in V(\text{MAP}^i)$, for every path p from n_0 to n in MAP^i , $\varphi^i(n) = \text{DEST}_G(\varphi(n_0), \lambda(p))$*

Proof. Straightforward from Theorem 2 □

To ease the notation, an homomorphism $\varphi : \text{MAP}^i \rightarrow G$ is denoted by φ^i .

Proof of theorem 2

We prove in the next Lemma the initial case of the induction.

Lemma 5.3 (Initial Case). *For any execution of Algorithm 1 on a graph G endowed with a binoculars labelling, MAP^1 is isomorphic to $B_G(v_0, 1)$.*

Proof. Since for every vertex $v \neq v_0$, $d(v_0, v) > 0$, during the first phase, the agent have to explore the first cluster composed by only one vertex, the home-base v_0 . Since the agent is initially located on v_0 , the agent only maps the neighbourhood of the homebase v_0 during this phase.

Initially, a first vertex n_0 is inserted at Line 2 into MAP^0 . This vertex identifies the home base v_0 . Let $\varphi(n_0) = v_0$. At Line 13, $B_G(v_0, 1)$ is gathered into $\mathcal{B}[n_0]$. Then, the agent updates its map (Line 15).

- First, since G has an injective port numbering, there is a unique pre-vertex (n_0, p, q) inserted into **PRE-VERT**, for every edge $v_0 w$ in $E(B_G(v_0, 1))$ labelled (p, q) .
- Remark that there is exactly one pre-vertex per equivalence classes of pre-vertices. Thus, we get that $\text{PRE-VERT}/\equiv^* \simeq \text{PRE-VERT}$.
- Consequently, since one vertex is added into MAP^1 for each equivalence class, there is a bijection between $V(\text{MAP}^1)$ and $V(B_G(v_0, 1))$.
- Moreover, since there is also one vertical edge inserted at Line 29 for each equivalence class, there is a bijection between vertical edges in $V(\text{MAP}^1)$ and vertical edges in $V(B_G(v_0, 1))$.
- It remains to prove that there is also a bijection between horizontal edge of $E(\text{MAP}^1)$ and horizontal edge of $E(B_G(v_0, 1))$.
- For every "horizontal" edge $ww' \in E(B_G(v_0, 1))$ i.e., $w' \neq v_0 \neq w$, there are $n_0[(n_0, \delta_{v_0}(w))], n_0[(n_0, \delta_{v_0}(w'))] \in E(\text{MAP})$ from the previous case. Moreover, the couple $\left(n_0, \delta_{v_0}(w), \delta_{v_0}(w'), (\delta_w(w'), \delta_{w'}(w))\right)$ is inserted into **Hor** at Line 23.
- Thus, an edge $\overline{[(n_0, \delta_{v_0}(w))][(n_0, \delta_{v_0}(w'))]}$ labelled by $(\delta_w(w'), \delta_{w'}(w))$ is inserted into $E(\text{MAP})$ at Line 31 if and only if there is an edge ww' in G labelled by $(\delta_w(w'), \delta_{w'}(w))$.

□

Consequently, let us define the homomorphism $\varphi^1 : \text{MAP}^1 \rightarrow G$ such that $\varphi^1(n) = \text{DEST}_G(v_0, \delta_{n_0}(n))$, for every $n \in V(\text{MAP}^1)$. Since φ^1 is an isomorphism (Lemma 5.3), we get that at phase 1, Theorem 2 is proved. Moreover, we get the following Corollary,

Corollary 5.4. *At phase 1, for every vertex $n \in V(\text{MAP}^1)$, for every path p from n_0 to n in MAP^1 , $\varphi^1(n) = \text{DEST}_G(\varphi(n_0), \lambda(p))$*

Phase i-1: (Induction hypothesis)

Suppose that the agent ends phase $i-1 > 0$ and the agent has computed MAP^{i-1} such that no error is detected. Moreover, suppose that there is an homomorphism $\varphi^{i-1} : \text{MAP}^{i-1} \rightarrow G$ define as follows:

- If $m \in V(\text{MAP}^{i-2})$, $\varphi^{i-1}(m) = \varphi^{i-2}(m)$
- If $m \in V(\text{MAP}^{i-1} \setminus \text{MAP}^{i-2})$,
 - Let $(n, p) \in \text{PRE-VERT}$ such that $\overline{[(n, p)]} = m \in V(\text{MAP}^{i-1})$.
 - Let $\varphi^{i-1}(m) = \text{DEST}_G(\varphi^{i-1}(n), \delta_n(m))$

Note that by induction and since $\text{MAP}^{i-2} \subseteq \text{MAP}^{i-1} \subseteq \text{MAP}^i$, for every $n \in V(\text{MAP}^{i-2})$, $\varphi^{i-1}(n)$ is well defined in MAP^i . The following corollary explains that the image in G of a vertex in MAP^{i-1} via φ^{i-1} is independent of the path followed by the agent.

Corollary 5.5. *At phase $i-1$, for every vertex $n \in V(\text{MAP}^{i-1})$, there is a vertex $w \in V(G)$ such that for every path p from n_0 to n in MAP^{i-1} , $\varphi^{i-1}(n) = \text{DEST}_G(\varphi^{i-1}(n_0), \lambda(p))$*

We suppose that Theorem 2 is proved at phase $i-1$, that is, φ^{i-1} is locally injective from MAP^{i-1} and locally surjective from $\text{MAP}^{i-1} \setminus \partial\text{MAP}^{i-1}$.

Remark that for every vertex $n \in \text{MAP}^{i-1} \setminus \partial\text{MAP}^{i-1}$ (n already explored at phase $i-1$), $B_{\text{MAP}^{i-1}}(n, 1) \simeq B_G(\varphi^{i-1}(n), 1) \simeq \text{getBino}(n)$.

Phase i:

We prove that when the agent ends phase $i > 1$ and has computed MAP^i , there is an homomorphism $\varphi^i : M^i \rightarrow G$ such that

- for every $n \in V(\text{MAP}^i)$, $\varphi^i|_{N_{M^i}(v)}$ is injective
- for every $n \in V(\text{MAP}^i \setminus \partial\text{MAP}^i)$, $\varphi^i|_{N_{M^i}(v)}$ is surjective

Next Lemma prove that the relation \equiv gathers in a same equivalence class the maximum set of "vertical" edges linking a same vertex in G . Moreover, it ensure that the image of a vertex via φ^i as defined above is independent of the choice of the representative $[n, p]$.

Remark 5.6. *Note that in some graphs which are not Weetman, some vertices can be duplicated in MAP .*

Lemma 5.7. *For every phase i of Algorithm 1, for every pre-vertices $(n, p), (m, q) \in \text{PRE-VERT}^i$, if $(n, p) \equiv^* (m, q)$ then there is a vertex $w \in B_G(\varphi^{i-1}(n), 1) \cap B_G(\varphi^{i-1}(m), 1)$ such that $\varphi^{i-1}(n)w, \varphi^{i-1}(m)w \in E(G)$ and $\delta_{\varphi^{i-1}(n)}(w) = p$ and $\delta_{\varphi^{i-1}(m)}(w) = q$.*

Proof. Suppose that there are two pre-vertices $(n, p), (m, q) \in \text{PRE-VERT}^i$ such that $(n, p) \equiv (m, q)$.

- From the algorithm, n and m are visited during phase i .
- Moreover, there is a vertex $\overline{[(n, p)]} = \ell \in V(\text{MAP}^i)$ linked to n and m such that $\delta_n(\ell) = p$ and $\delta_m(\ell) = q$.
- By induction, there is $u, v \in V(G)$ such that $u = \varphi^{i-1}(n)$, $v = \varphi^{i-1}(m)$, and $\varphi^{i-1}(nm) = \varphi^{i-1}(n)\varphi^{i-1}(m)$ is an edge in $E(G)$.
- Moreover, let $p_n : n_0 \rightarrow n$ (resp $p_m : n_0 \rightarrow m$) denotes the path follows by the agent from the beginning of the execution when it visits n (resp. m) for the first time.
- By induction, $\varphi^{i-1}(n) = \text{DEST}_G(v_0, \lambda(p_n)) = u$ and $\varphi^{i-1}(m) = \text{DEST}_G(v_0, \lambda(p_m)) = v$, that is, u and v are vertices where the agent is located corresponding to n and m in its map.
- Since $(n, p), (m, q) \in \text{PRE-VERT}^i$ and from homomorphism definition, the agent has seen two times the triangle uvw . Once inside $B_G(\varphi^{i-1}(n), 1)$ when it visits n such that $\psi(n) = u$ and once inside $B_G(\varphi^{i-1}(m), 1)$ when it visits m such that $\psi(m) = v$.
- Consequently, $w \in B_G(\varphi(n), 1) \cap B_G(\varphi(m), 1)$ and there is a vertex $w \in V(G)$ such that $\varphi^i(u)w\varphi^i(m)$ is a triangle in $E(G)$, we get the first case of this Lemma.

We now prove the case $(n, p) \equiv^*(m, q)$.

- By definition, $(n, p) \equiv^*(m, q)$ implies that there is a sequence of pre vertices $(n_1, p_1), \dots, (n_k, p_k) \in \text{PRE-VERT}$ such that
 - $(n, p) = (n_1, p_1)$ and $(m, q) = (n_k, p_k)$
 - $\forall 1 \leq h < k, (n_h, p_h) \equiv (n_{h+1}, p_{h+1})$
- Moreover, by induction, $\varphi^{i-1}(n_1, \dots, n_k) = \varphi^{i-1}(n_1) \dots \varphi^{i-1}(n_k)$ is a path in G .
- From the previous case, for every $(n_h, p_h) \equiv (n_{h+1}, p_{h+1})$, there is $w_h \in V(G)$ such that
 - $\varphi^{i-1}(n_h)\varphi^{i-1}(n_{h+1})w_h \subset G$ is a triangle
 - $\delta_{\varphi^{i-1}(n_h)}(w_h) = p_h$ and $\delta_{\varphi^{i-1}(n_{h+1})}(w_h) = p_{h+1}$.
- From the transitivity of \equiv and the injective port numbering function of G , we get that $\delta_{\varphi^{i-1}(n_h)}(w_h) = p_{h+1} = \delta_{\varphi^{i-1}(n_{h+1})}(w_{h+1})$.
- Consequently, we get that $w_h = w_{h+1}$ for every $1 \leq h < k$.
- We prove that there is a unique vertex $w \in V(G)$ such that $\delta_{\varphi^{i-1}(n)}(w) = p$ and $\delta_{\varphi^{i-1}(m)}(w) = q$.

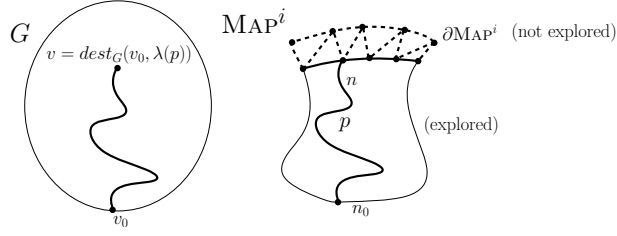
□

The above Lemma permits us to define the homomorphism $\varphi^i : V(\text{MAP}^i) \rightarrow V(G)$ such that for every $m \in V(\text{MAP}^{i-1})$, $\varphi^i(m) = \varphi^{i-1}(m)$ and for every $m \in V(\text{MAP}^i \setminus \text{MAP}^{i-1})$, $\varphi^i(m) = \text{DEST}_G(n, p)$ such that $m = \overline{[(n, p)]}$.

It is straight forward to prove that φ^i is well defined for vertices. So, we prove in the next lemma that $\varphi^i : \text{MAP}^i \rightarrow G$ is an homomorphism, that is, the image of an edge is an edge.

Lemma 5.8. *At phase i , for every edge $nm \in E(\text{MAP}^i)$, $\varphi^i(nm) \in E(G)$*

Proof. First, we prove the lemma for vertical edges and then, for horizontal edges.



Vertical edges.

- We know that for every edge $nm \in E(\text{MAP}^i)$ such that $n \notin V(\partial \text{MAP}^i)^i$ and $m \in V(\partial \text{MAP}^i)^i$ (vertical 'edge'), there is a pre-vertex $[n, p] \in \text{PRE-VERT}^i$ such that $\overline{[(n, p)]} = m$.
- Moreover, there is two vertices $u, v \in V(G)$ such that $\varphi^i(m) = v \neq \varphi^i(n) = u$ and $\delta_u(v) = p$.
- we get that $\varphi^i(nm) = \varphi^i(n)\varphi^i(m) = uv \in E(G)$ is well defined for every "vertical" edge $nm \in E(\text{MAP}^i)$

Horizontal edges. Now, we prove that φ^i correctly maps "horizontal" edges in MAP to G .

- By construction, every edge $\overline{[(n, p)]}[\overline{(m, q)}] \in E(\text{MAP}^i \setminus \text{MAP}^{i-1})$ implies that there is $(n, p, p', (r, s)) \in \text{Hor}$ such that $(n, p), (n, p') \in \text{PRE-VERT}$ are two pre-vertices and $(n, p) \not\equiv (n, p') \equiv (m, q)$ (injective port numbering function of G).
- Moreover, the agent located on $u = \varphi^{i-1}(n)$ has seen a triangle $uw' \in B_G(\varphi^{i-1}(n), 1)$ such that
 - the edge ww' is labelled (r, s)
 - there is no $m, m' \in V(B_{\text{MAP}^{i-1}}(n, 1))$ such that $\delta_n(m) = p$ and $\delta_n(m') = p'$
- From the previous case, $w = \varphi^i(\overline{[(n, p)]})$ and $w' = \varphi^i(\overline{[(m, q)]})$.
- Moreover, since $p \neq p', w \neq w'$ and thus, $\varphi^i(n[\overline{(n, p)}]) = vw \neq \varphi^i(n[\overline{(n, p')}] = vw'$.
- Thus, $\varphi^i(\overline{[(n, q)]}[\overline{(m, q)}]) = \varphi^i(\overline{[(n, q)]})\varphi^i(\overline{[(m, q)]}) = ww' \in V(G)$ is well defined for every horizontal edge $\overline{[(n, q)]}[\overline{(m, q)}]$ in $E(\text{MAP}^i)$

□

By induction, Theorem is already proved for every vertex included in MAP^{i-1} which are not explored phase i . Consequently, we only prove Theorem 2 for

- vertices explored phase i (Lemma 5.9 and 5.10), i.e., that belongs to $\partial \text{MAP}^{i-1} \setminus \partial \text{MAP}^i$
- and newly discovered vertices (Lemma 5.11), i.e., that belongs to $\text{MAP}^i \setminus \text{MAP}^{i-1}$

Lemma 5.9. *for every $n \in V(\text{MAP}^i \setminus \partial \text{MAP}^i)$, for every edges $m_1 m'_1, m_2 m'_2 \in E(B_{\text{MAP}^i}(n, 1))$, if $m_1 m'_1 \neq m_2 m'_2$ then $\varphi(m_1 m'_1) \neq \varphi(m_2 m'_2)$.*

$B_{\text{MAP}}(n, 1)$

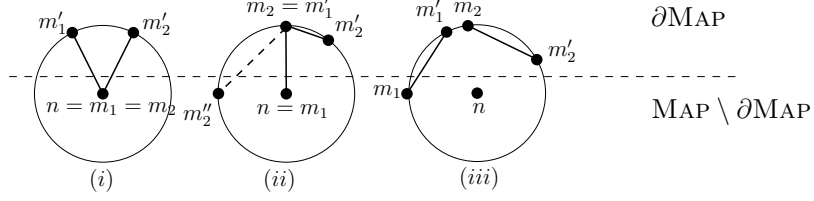


Fig. 5: Different cases of Lemma 5.9

Proof. Let $n \in V(\text{MAP}^i \setminus \partial \text{MAP}^i)$ be a vertex explored phase i and let $m_1 m'_1, m_2 m'_2 \in E(B_{\text{MAP}^i}(n, 1))$ such that $m_1 m'_1 \neq m_2 m'_2$. Three cases appear,

- i) $m_1 = m_2 = n$ and $m'_1 \neq m'_2$.
 - Since $n \in V(\text{MAP}^i \setminus \partial \text{MAP}^i)$, by induction, $\mathcal{B}[n] \simeq B_G(\varphi^i(n), 1)$
 - Since G and MAP^i have injective ports labellings, there is two different vertices $w \neq w' \in V(G)$ such that $w = \varphi^i(m'_1) \neq w' = \varphi^i(m'_2)$
 - Thus, since φ^i is an homomorphism, $\varphi^i(w_1 w'_1) \neq \varphi^i(w_2 w'_2)$
- ii) $m_1 = n$ and $m'_1 = m_2$
 - First, remark that by definition, for every vertex $m \in V(B_{\text{MAP}^i}(n, 1))$, there is an edge $nm \in E(\text{MAP}^i)$.
 - So there are $nm'_1, nm'_2 \in E(\text{MAP}^i)$,
 - From the previous case we know that $\varphi^i(nm'_1) \neq \varphi^i(nm'_2)$.
 - So $\varphi^i(m'_1) \neq \varphi^i(m'_2)$ and $\varphi^i(m_1 m'_1) \neq \varphi^i(m_2 m'_2)$
- iii) $n \neq m_1 \neq m_2 \neq n$
 - From the previous case $\varphi^i(w_1) \neq \varphi^i(w_2)$ and $\varphi^i(w'_1) \neq \varphi^i(w'_2)$ in G .
 - Since φ^i is an homomorphism, $\varphi^i(w_1 w'_1) \neq \varphi^i(w_2 w'_2)$

□

Lemma 5.10. *For every $n \in V(\text{MAP}^i \setminus \partial \text{MAP}^i)$, $\varphi|_{N_{\text{MAP}^i}}(n)$ surjective*

Proof. – Since φ^i is an homomorphism, every adjacent vertex of n has an image via φ^i .
– Moreover, from Lemma 5.9, φ^i is locally injective from $B_{\text{MAP}^i}(n, 1)$ to $B_G(\varphi^i(n), 1)$.
– Consequently, every adjacent vertex of n has a unique image via φ^i .
– Finally, $\mathcal{B}[n] \simeq B_{\text{MAP}^i}(n, 1)$ ensures that we map in MAP , every vertex present in $B_G(\varphi^i(n), 1)$ which is isomorphic by induction to $\mathcal{B}[n]$.
– So, φ^i is locally surjective for every $n \in V(\text{MAP}^i \setminus \partial \text{MAP}^i)$ explored phase i .

□

Lemma 5.11. *for every $n \in V(\partial \text{MAP}^i)$, $\varphi|_{N_{\text{MAP}^i}^i}(n)$ injective*

Proof. For every phase i , for every $n \in \partial \text{MAP}^i$, two case appears:

If $n \in V(\partial \text{MAP}^{i-1})$, then n is not explored phase i and by induction, Lemma is proved.

If n is inserted into MAP at phase i , that is, $n \in V(\partial \text{MAP}^i \setminus \partial \text{MAP}^{i-1})$, then for every neighbours $m \neq m'$ of n in $V(\text{MAP}^i)$, two cases appear:

- Either, m and m' in $V(\text{MAP}^i \setminus \partial \text{MAP}^i)$, and in this case, $\left((m, \delta_m(n)), (m', \delta_{m'}(n))\right) \in \equiv^*$. By induction, $\varphi^i(m) \neq \varphi^i(m')$.
- Either, m and m' belong to ∂MAP^i .
 - In such a case, since there is always a vertex ℓ (resp. ℓ') such that $(\ell, \delta_\ell(n), \delta_\ell(m), \lambda(nm)) \in \text{Hor}$ (resp. $(\ell', \delta_{\ell'}(n), \delta_{\ell'}(m'), \lambda(nm')) \in \text{Hor}$)
 - we get that $[\ell, \delta_\ell(m)] \neq [\ell, \delta_\ell(n)] = [\ell', \delta_{\ell'}(n)] \neq [\ell', \delta_{\ell'}(m')]$.
 - From previous case, $\varphi^i(\ell) \neq \varphi^i(\ell')$.
 - Since MAP^i has an injective port labelling, $\delta_n(m) \neq \delta_n(m')$.
 - So $\varphi^i(m) \neq \varphi^i(m')$ since otherwise, W.l.o.g. $\delta_n(m) \neq \delta_{\varphi(n)}(\varphi(m))$.

□

We proved the Theorem 2.

It remains to prove that φ^i preserves triangles in order to prove the correctness of the map construction.

Lemma 5.12. *for every vertex $n \in V(\text{MAP}^i)$ explored phase i , for every triangle $v_1v_2v_3 \subseteq B_G(\varphi^i(n), 1)$, there is a triangle $m_1m_2m_3 \subseteq B_{\text{MAP}^i}(n, 1)$ such that $\varphi(m_h) = v_h$ for every $1 \leq h \leq 3$ and $\varphi(m_1m_2m_3) = v_1v_2v_3$*

Proof. for every $n \in V(\partial \text{MAP}^{i-1} \setminus \partial \text{MAP}^i)$ and for every triangle $v_1v_2v_3 \subseteq B_G(\varphi^i(n), 1)$,

- From Lemma 5.9, there is $m_1m_2m_3 \in V(B_{\text{MAP}^i}(n, 1))$ that $\varphi^i(m_1) \neq \varphi^i(m_2) \neq \varphi^i(m_3)$ and $\varphi(m_1)\varphi(m_2) \neq \varphi(m_2)\varphi(m_3) \neq \varphi(m_3)\varphi(m_1)$
- Since we ensure at line 32 that $B_{\text{MAP}^i}(n, 1) \simeq B_G(\varphi(n), 1)$, we get that $m_1m_2m_3$ is a triangle in MAP if and only if $\varphi^i(m_1)\varphi^i(m_2)\varphi^i(m_3)$ is a triangle in G .

□

Thus, from Lemma above and from Theorem 2, we get the next corollary,

Corollary 5.13. *If, at the phase i , the agent halts its exploration (without error), then MAP^i is a simplicial covering of G*

So we finish this proof by the following Theorem,

Theorem 3. *If the agent halts its exploration, then the graph is explored*

Proof. – First, if the agent halts its exploration then it does not find any mistake in its map (Line 32).

- Moreover, it explores all vertices in its map.

- Suppose the agent halts phase i . we get that $\partial \text{MAP}^i = \emptyset$ (all vertices explored) and thus, MAP^i is locally bijective (injective + surjective).
- $\Rightarrow \varphi$ is a covering
- \Rightarrow (surjective covering) $|\text{MAP}^i| > |G|$
- $\Rightarrow G$ is explored

□

This Theorem means that the agent cannot halts before exploring every vertex of any graph. To conclude this part, we have to prove that the agent always halts on Weetman graphs.

Theorem 4. *For every graph $G \in \text{Weetman}$, the algorithm explores G and halts*

Proof. First, note that if the agent halts its execution on a Weetman graph G , from Lemma 4.1 and Lemma 5.13, MAP is isomorphic to G . Moreover, since G satisfied the Interval Condition, every vertex v of $V(G)$ has a unique image n in $V(\text{MAP})$. Since G satisfied the Triangle Condition, every edge $\varphi(n)\varphi(m)$ in $E(G)$ has a unique corresponding image nm in $V(\text{MAP})$. Consequently, no error can be found in any execution of \mathcal{A} on a Weetman graph G . Finally, since the clusters of G form a tree and since the agent performs a DFS over clusters, the agent will reach a phase where all of the nodes in its map are explored. □

Remark 5.14. *Let u, v, w be a triple of vertices in G and let $n = \varphi^{i-1}(u)$, $m = \varphi^{i-1}(v)$ and $l = \varphi^{i-1}(w)$ be one pre image of u, v, w in MAP and $d(v_0, w) > d(v_0, u) = d(v_0, v)$. If u, v, w does not respect the Interval condition from v_0 in G , then the "top" vertex w will be duplicated in MAP . In fact, since there is no sequence of adjacent triangles $v_1v_2w, \dots, v_kv_{k+1}w$ in G explored in the same phase, the agent has no enough pre-vertices relation (\equiv) at the end of phase to gather $(n, \delta_u(w)), (m, \delta_u(w))$ in a same equivalence class. Figure 6 illustrates such an error in MAP .*

Remark 5.15. *So, for every phase i , if two vertices in MAP are linked to a same vertex in ∂MAP^i , then we know that there corresponding vertex in G are also linked together with a third vertex which is newly discovered. From the previous remark, we know that we can duplicate a vertex w of G in MAP . But in this case, every vertical edge linking w in G are partitioned and distributed over every "copies" of w in MAP (not duplicated).*

5.1 Complexity

Let G be a Weetman graph and let v be the homebase of the agent. Remark that the total number of clusters $|\mathcal{CIR}(G)|$ is bounded by $|V|$, $\sum_{C \in \mathcal{CIR}_G(v)} |V(C)| = |V(G)|$.

Note that since in a tree T , $E(T) = V(T) - 1$, exploring a tree takes at most $2|E(C)| \leq 2|V(C)|$ steps even if the agent has to come back to the roots. Moreover, exploring a spanning tree in a graph is a worst case since a cycle in

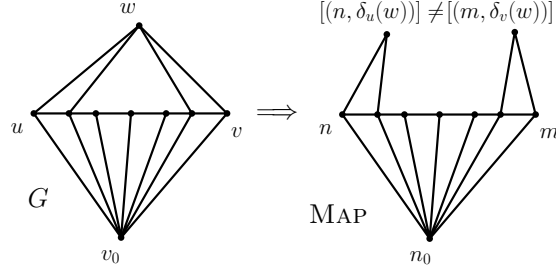


Fig. 6: $IC(v_0)$ not satisfied on u, v, w

the course of the agent implies that at least one backtracking of the agent is avoided (in front of the spanning tree exploration), which decreases the number of moves.

Since every edge in a cluster is a horizontal edge, the agent crosses at most $\sum_{C \in \mathcal{CIR}(G)} 2|V(C)| \leq 2|V|$ horizontal edges to explore every cluster.

Horizontal edges in C can be cross two times more. Once the agent goes to the vertex in C which permits it to reach the next cluster C' to visit. Once when the agent has finished the exploration of the "branch" starting at C' and backtracks to C to go to the next cluster C'' to visit. Note that clusters have to be ordered in a way that the agent can go from one to another in a $\mathcal{O}(|V(G)|)$ moves, that is, in a linear number of moves. Moreover, a DFS ordering ensures that once a "branch" is explored, the agent never returns in this branch. Consequently, we prove that every horizontal edges is crossed by the agent a linear number of times in an execution.

Since $\mathbf{Cluster}(G)$ is a tree and the agent performs a DFS on the clusters, the agent crosses at most two vertical edges per clusters. We get that the agent crosses a linear number of times vertical edges to go from one to another cluster in an execution.

Since the agent explores a spanning tree of G , we get that the number of edges crossed is bounded by the number of vertices explores.

Consequently, since we prove that every edge crossed by the agent a linear number of times, the complexity of Algorithm 1 is achieved in $\mathcal{O}(|V(G)|)$ moves.

We get our final theorem,

Theorem 5. *Algorithm 1 explores and computes a map of Weetman graphs with a number of moves that is linear in the size of the graph.*

6 Conclusion

We have presented an algorithm that explores all chordal graphs in a linear number of moves by a mobile agent using binoculars to see the edges between nodes adjacent to its location. The main contribution is that the exploration is

fast and, contrary to previous works, the agent does not need to know the size or the diameter (or bounds) to halt the exploration. Using binoculars permits to not visit every edge while still being assured of having seen all nodes, which is usually not possible at all without binoculars without additional information about some graph parameters.

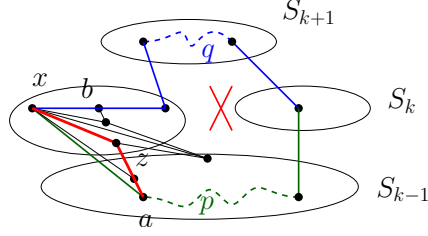
We have actually used properties of chordal graphs that are verified in a larger class of graphs : the Weetman graphs. This class of graphs belongs to families of graphs that are defined by local metric properties and whose clique complexes are simply connected, which is a necessary condition for linear exploration without knowledge (see [CGN15]).

Known such families are the family of bridged graphs, or the family of dismantlable/cop-win graphs that have found numerous application in distributed computing. Chordal and bridged graphs are Weetman and Algorithm 1 also efficiently explores these graphs. Dismantlable graphs are not necessarily Weetman, but given that cop-win graphs can also be defined by an elimination order, a very interesting open question would be to prove, or disprove, that there is a linear Exploration algorithm for cop-win graphs.

References

- AGP⁺11. C. Ambühl, L. Gąsieniec, A. Pelc, T. Radzik, and X. Zhang. Tree exploration with logarithmic memory. *ACM Trans. Algorithms*, 7(2):17:1–17:21, Mar 2011.
- AKL⁺79. R. Aleliunas, R. M. Karp, R.J. Lipton, L. Lovász, and C. Rackoff. Random walks, universal traversal sequences, and the complexity of maze problems. In *FOCS*, pages 218–223, 1979.
- BH99. M. Bridson and A. Haefliger. *Metric Spaces of Non-positive Curvature*. Springer-Verlag Berlin Heidelberg, 1999.
- BV01. P. Boldi and S. Vigna. An effective characterization of computability in anonymous networks. In *DISC*, 2001.
- CFI⁺05. R. Cohen, P. Fraigniaud, D. Ilcinkas, A. Korman, and D. Peleg. Label-guided graph exploration by a finite automaton. In *ICALP*, volume 3580 of *LNCS*, page 335–346, 2005.
- CGM12. J. Chalopin, E. Godard, and Y. Métivier. Election in partially anonymous networks with arbitrary knowledge in message passing systems. *Distrib. Comput.*, 2012.
- CGN15. Jérémie Chalopin, Emmanuel Godard, and Antoine Naudin. Anonymous graph exploration with binoculars. In Yoram Moses, editor, *Distributed Computing*, volume 9363 of *Lecture Notes in Computer Science*, pages 107–122. Springer Berlin Heidelberg, 2015.
- Das13. Shantanu Das. Mobile agents in distributed computing: Network exploration. *Bulletin of EATCS*, 1(109), Aug 2013.
- GR08. Leszek Gąsieniec and Tomasz Radzik. Memory efficient anonymous graph exploration. In *WG*, volume 5344 of *LNCS*, page 14–29, 2008.
- Ilc08. David Ilcinkas. Setting port numbers for fast graph exploration. *Theoretical Computer Science*, 401(1–3):236–242, 2008.
- Kou02. Michal Koucký. Universal traversal sequences with backtracking. *Journal of Computer and System Sciences*, 65(4):717–726, 2002.

- LO99. Danny B. Lange and Mitsuru Oshima. Seven good reasons for mobile agents. *Commun. ACM*, 42(3):88–89, Mar 1999.
- LS77. R.C. Lyndon and P.E. Schupp. *Combinatorial group theory*. Ergebnisse der Mathematik und ihrer Grenzgebiete. Springer-Verlag, 1977.
- Rei08. O. Reingold. Undirected connectivity in log-space. *J. ACM*, 55(4), 2008.
- Ros00. Kenneth H. Rosen. *Handbook of Discrete and Combinatorial Mathematics*. Chapman & Hall/CRC, 2000.
- RT87. M.A. Ronan and J. Tits. Building buildings. *Mathematische Annalen*, 278(1-4):291–306, 1987.
- Wee94. Graham M. Weetman. A construction of locally homogeneous graphs. *Journal of the London Mathematical Society*, 50(1):68–86, 1994.
- YK96. M. Yamashita and T. Kameda. Computing on anonymous networks: Part i - characterizing the solvable cases. *IEEE TPDS*, 7(1):69–89, 1996.



A Appendix Section

From the homotopy relation on cycles, we get the following Proposition.

Proposition A.1. *Given a not contractible cycle c in a graph G and a cycle c' that is homotopic to c , then c' is not contractible.*

Theorem 6. *Clusters of a simply connected graph form a tree*

Proof. By contradiction, there is two vertices $x, y \in V(G)$ such that $x, y \in S^k$ and there is no path $p' \subset S^k$ from x to y . Moreover, to get a cycle of clusters, there is a path $q \subset G \setminus B^{k-1}$ from y to x . Note that, W.l.o.g, there is also a path $p : x \rightarrow y \subset V(G)$ such that $p \setminus \{xy\} \subset B^{k-1}$.

We denote by $C(x, y)$ a pair of paths $p : x \rightarrow y$ and $q : y \rightarrow x$ such that $p \setminus \{xy\} \subset B^{k-1}$, $q \subset B^{k+1}$ and $\mathcal{A}(C(x, y))$ is minimal.

Among every pair of vertices x', y' not relied by a path $p' \subset S^k$, let x, y be the couple minimising $\mathcal{A}(C(x, y))$.

Let $a \in p \cap S^{k-1}$ be the vertex such that $ax \in E(p)$ and let $b \in q$ such that $bx \in E(q)$. Remark that since $d(v_0, x) = k$, we get that $k \leq d(v_0, b) \leq k+1$

Since G is simply connected, there is a minimal disk diagram (D, f) for the cycle $C(x, y) = pq$. By definition, $f(\partial D) = C(x, y)$ and we denote by \tilde{x} the pre images of x in D . W.l.o.g, $f(\tilde{x}) = x \in V(G)$.

Since D is a planar triangulation, there is a path $\tilde{s} = \tilde{v}_1 \tilde{v}_2 \dots \tilde{v}_{\ell-1} \tilde{v}_\ell \subset V(D)$ such that for every $1 \leq i < \ell$, $f(\tilde{v}_1) = a, f(\tilde{v}_\ell) = b$ and $\tilde{x} \tilde{v}_i \tilde{v}_{i+1}$ is a triangle in D .

Since $d(v_0, a) = d(v_0, x) - 1 = k - 1$ and $k \leq d(v_0, b) \leq k + 1$, for every $1 \leq i \leq \ell$, $k - 1 \leq d(v_0, v_i) \leq k + 1$.

Consequently, among every $\tilde{v}_j \in \tilde{s}$, there is a unique vertex $\tilde{v}_i \in \tilde{s}$ such that $d(v_0, v_i) = k$ and for every $1 \leq h < i$, $d(v_0, v_h) = k - 1$.

Remark that there is a couple of paths $p' : v_i \rightarrow y$ and $q' : y \rightarrow v_i$ such that $p' = p \setminus \{ax\} \cup \{av_2 \dots v_i\}$, $q' = q \cup \{xv_i\}$. Moreover, since x and y are not relied by a path in S^k and since $v_i x \in E(S^k)$, v_i and y are also not relied by a path in S^k .

Since $p' \setminus \{v_i y\} \subset B^{k-1}$, and $q' \subset G \setminus B^{k-1}$, it remains to prove that $\mathcal{A}(p'q')$ is smaller than $\mathcal{A}(pq)$.

Since $D' = D \setminus \{uv_h, \forall 1 \leq h < i\}$ is a disk diagram for $p'q'$, it is easy to see that $\mathcal{A}(D') < \mathcal{A}(D)$ since for every $1 \leq h < i$, the triangle $\tilde{u} \tilde{v}_h \tilde{v}_{h+1}$ does not appear in D' .

We get a contradicton on the choice of x, y . Consequently, there is no couple of vertices inside S^k , for every k , which are not relied by a path inside S^k . We prove that there is no cycle of clusters in simply connected graph and thus, clusters form a tree. \square